Programming 2
Tutorial and Practical 12

This week's tutorial and practical sessions cover recursion and binary search trees.

Tutorial Session

Recursion

1. Revisit the simple examples of recursion discussed in the lecture:

- Multiplication
- Factorial
- String length
- String reversing

Follow through these algorithms step-by-step, and ensure you understand:

a) How the recursive algorithm arrives at a result; and

b) Why the recursive steps and terminating conditions were chosen.

2. Consider the process of running through every subdirectory (and subsequent subdirectories) on a hard disk. How can this be solved using recursion?

Binary Search Trees

3. How does a tree structure differ from a list?

4. Are all binary search trees inherently balanced? Why/why not?

5. Why is deletion significantly more difficult than addition or retrieval? What special cases must be considered?

Practical Session

1. Look up the file handling sections of the API (check the `File` class) and write a program to display a directory tree using a recursive directory traversal as mentioned in the tutorial.

Optional Advanced Exercise

2. Write a binary search tree that has functionality equivalent to the linked list created in last week's laboratory class. (If properly written, it should work with the same simple test driver that you implemented last week.) To review the requirements:

Phone numbers are to be uniquely identified by the name, and as such duplicate names should never be stored. The following functionality should be present:

- *Add data:* given a phone number object, it should store it (after checking that there isn't already a matching number). Data should be kept in sort order.
- *Retrieve data:* given a name, it should retrieve the appropriate object.
- *Delete data:* given a name, it should delete the appropriate object, if it exists.
- *Summarise data:* create an array of objects based on the data in the list, and return it.